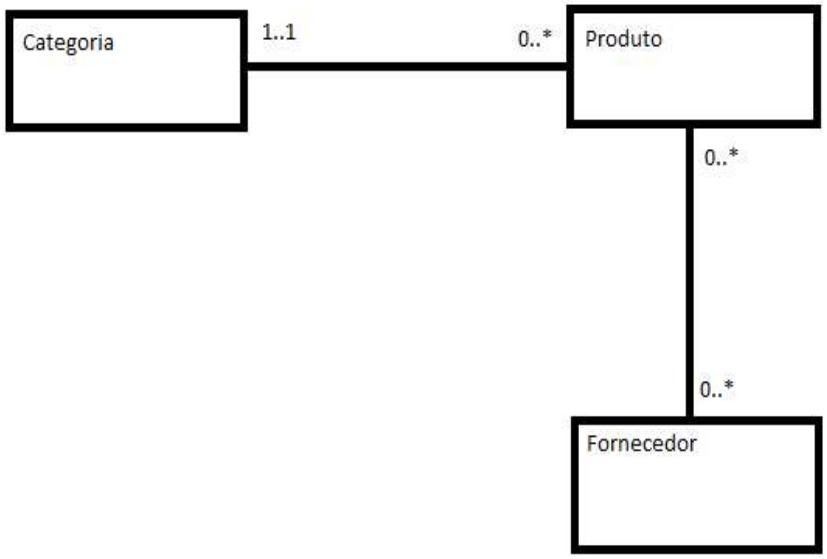
Fatec Praia Grande

Paulo R. T. Cândido

NodeJS - parte 3

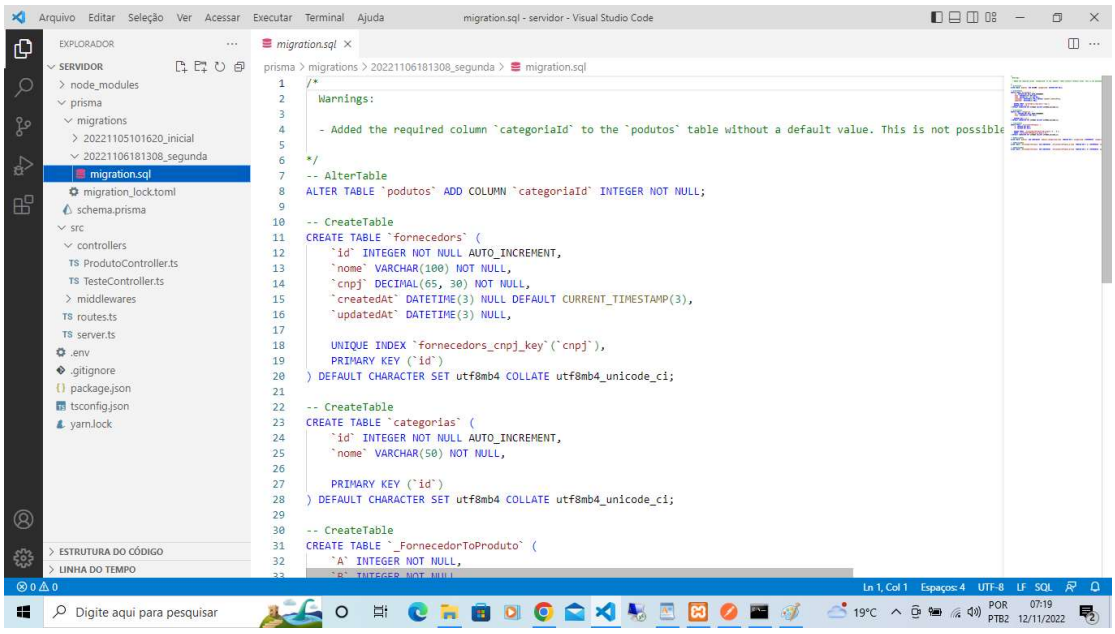
Models associados

1) Definir os models



2) Executar migrações

yarn prisma migrate dev --name segunda



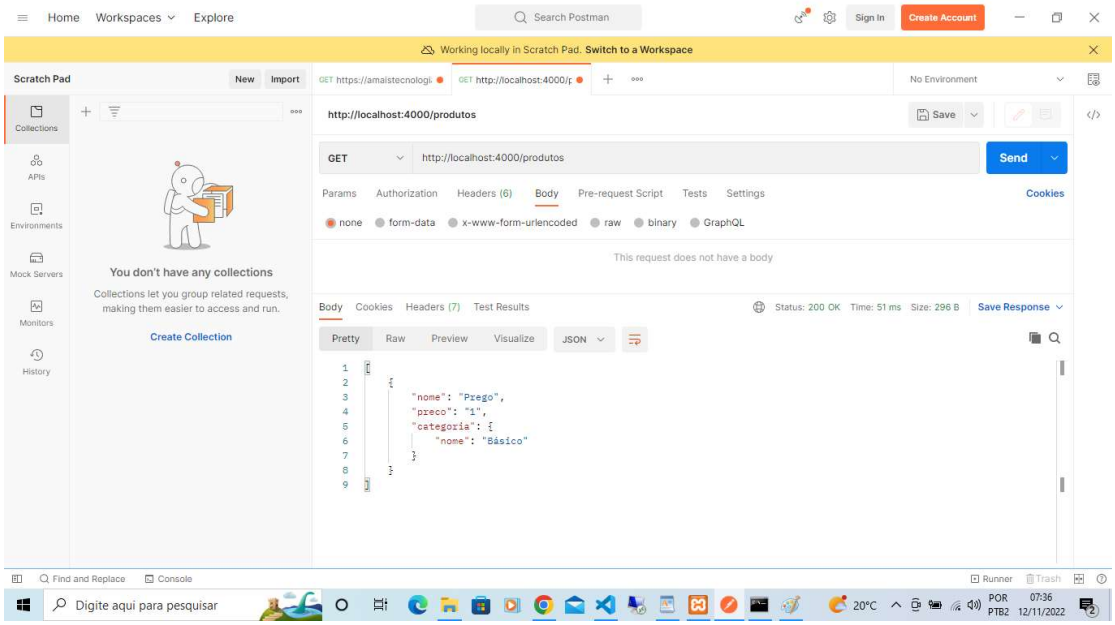
3) Alterar controller de produtos

Função index

```
async index(req:Request,res:Response){
   const prisma = new PrismaClient();
   const produtos = await prisma.produto.findMany( // recupera todos os produto
     {
       orderBy:{nome:'asc'},
       select:{
         nome:true,  // seleciona as propriedade desejadas de Produto
         preco:true,
         categoria:{
           select:{nome:true}  // traz do model relacionado Categoria apenas o nome
```

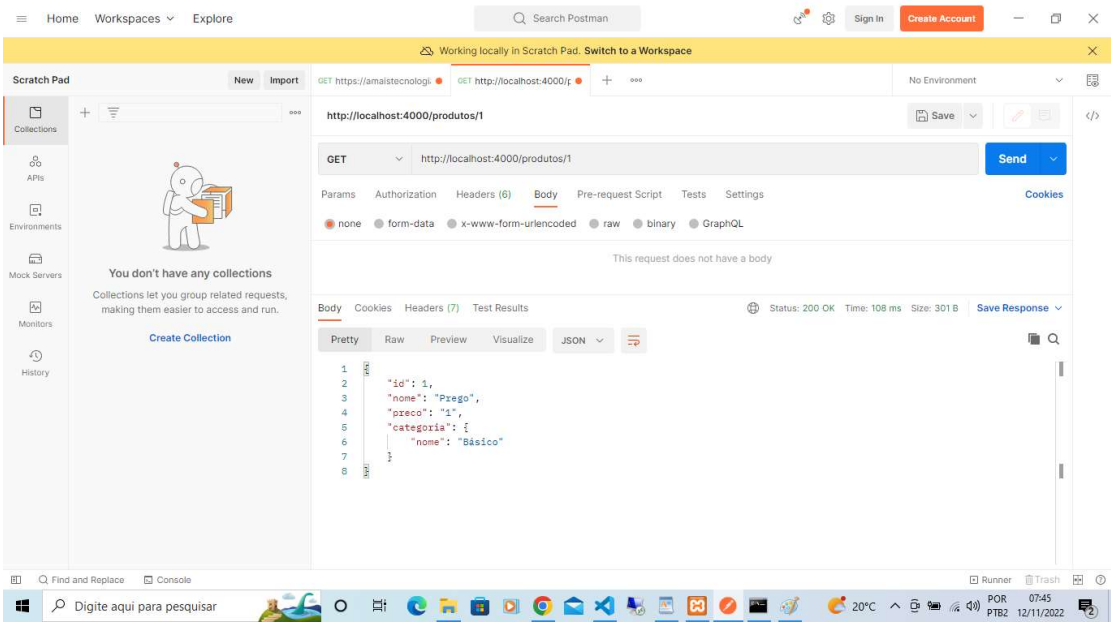
```
            }
          }
        }
    );
    res.status(200).json(produtos);
}
```

Obs: para o teste abaixo foi cadastrado ditetamente no banco de dados categoria com id=1 e nome="Básico"



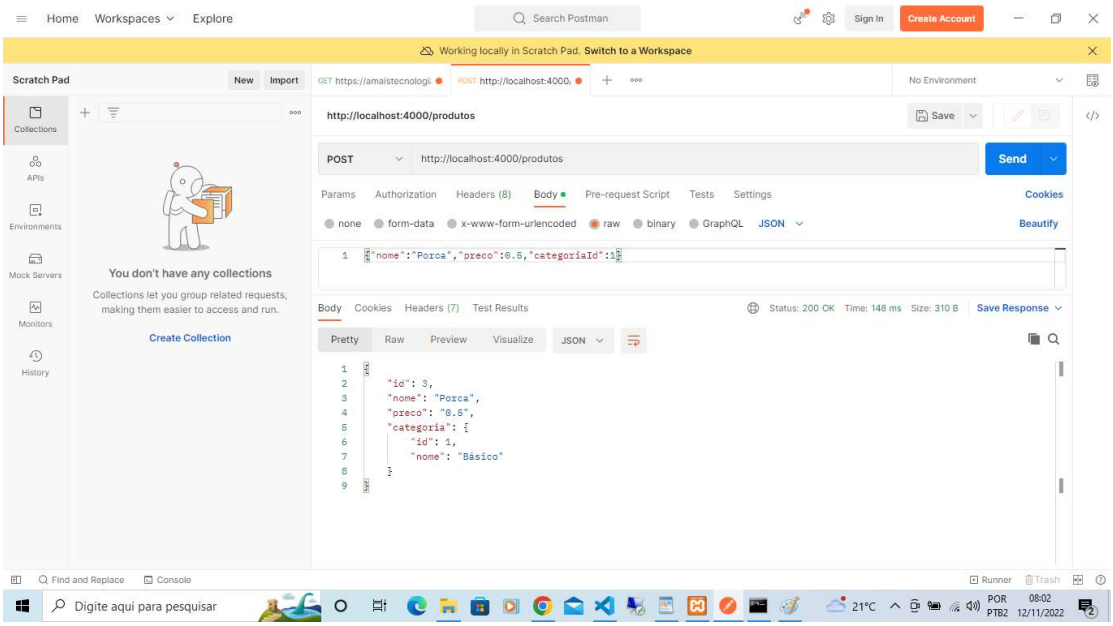
Função show

```
async show(req:Request,res:Response){
    const prisma = new PrismaClient();
    const produto = await prisma.produto.findUnique( // busca produto conforme where
        {
            where:{id: Number(req.params.id)},
            select:{
                id: true,  // seleciona as propriedade desejadas de Produto
                nome:true,
                preco:true,
                categoria:{
                    select:{nome:true}  // traz do model relacionado Categoria apenas o nome
                }
            }
        }
    );
    res.status(200).json(produto);
}
```



Função Store

```
async store(req:Request,res:Response){
    const prisma = new PrismaClient();
    //obtém json vindo do cliente. Exemplo Formato: {nome: "Prego", preco:2.3, categoriaId:1}
    const {nome, preco, categoriaId} = req.body;
    const novoPoduto = await prisma.produto.create(
        {
            data: {
                nome: nome,
                preco: preco,
                categoria: {connect:{id:categoriaId}} // associa produto à categoria
            },
            select: {
                id:true,
                nome:true,
                preco:true,
                categoria: true  // traz todos os dados de catedoria
            }
        }
    );
    res.status(200).json(novoPoduto);
}
```

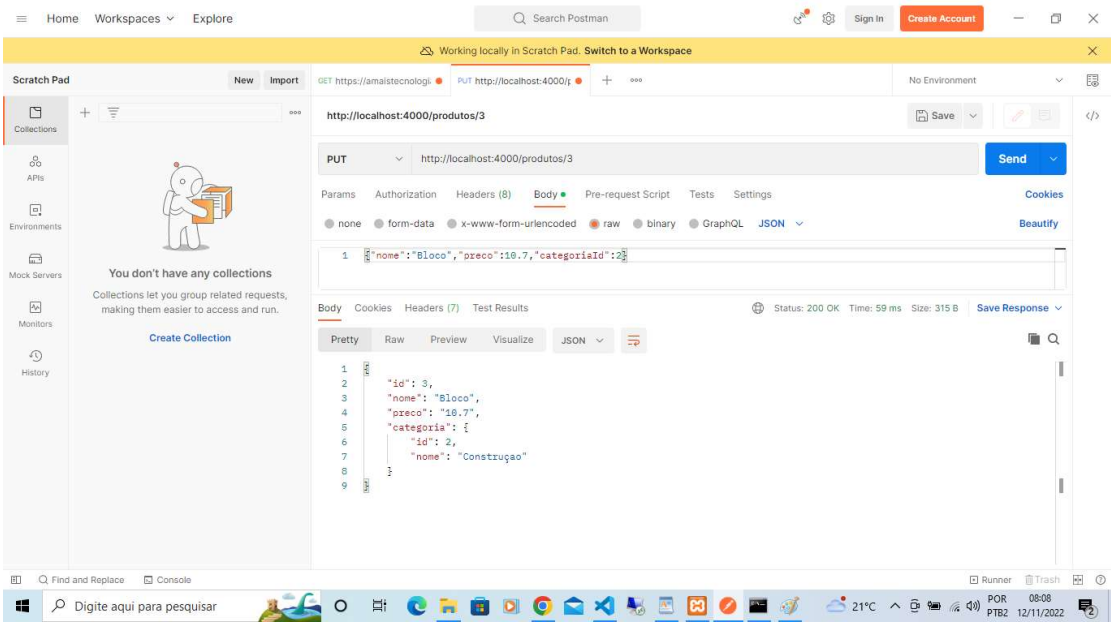


Função update

Obs.: para funcionar o teste abaixo é preciso inserir categoria id=2, nome="Construção"

```
async update(req:Request,res:Response){
    const prisma = new PrismaClient();
    const {nome, preco, categoriaId} = req.body;
    const produtoAlterado = await prisma.produto.update(
        {
            where: {id: Number(req.params.id) },
            data: {
                nome: nome,
                preco: preco,
                categoria: {connect:{id:categoriaId}} // associa produto à categoria
            },
            select: {
                id:true,
                nome:true,
                preco:true,
                categoria:true
            }
        }
    );
    res.status(200).json(produtoAlterado);
}
```

No exemplo é alterado o produto com id=3, informado na URL.



Função delete não precisa ser alterada.

4) Associar produtos a fornecedores

```
async associarFornecedores(req:Request,res:Response){
    // Exemplo de json recebido: {fornecedores:[1,2]}
    const {fornecedores} = req.body;
    const dados = fornecedores.map( (x:any)=>{return {id:x}} ); // resulta: [{id:1},{id:2}]
    const prisma = new PrismaClient();
    const produtoAlterado = await prisma.produto.update(
      {
        where: {id: Number(req.params.id) },
        data: {
          fornecedores: {connect:dados} // associa produto à categoria
        },
        select :{
          nome: true,
          preco: true,
          categoria: true,
          fornecedores: true
        }
      }
    );
    return res.status(200).json(produtoAlterado);
  }
```

Cria rota

Obs.: para este teste, cadastrar 2 fornecedores diretamente no banco de dados